Bu defterde tarihlerle çalışacağız.

Hadi başlayalım!

Get our environment set up[¶](https://www.kaggle.com/code/alexisbcook/parsing-dates#Get-our-environment-set-up)

Kullanacağımız kütüphaneleri ve veri setini yüklemek ilk yapmamız gereken şey. 2007 ve 2016 yılları arasında meydana gelen toprak kaymaları hakkında bilgi içeren bir veri seti ile çalışacağız. Bir sonraki alıştırmada, yeni öğrendiğiniz becerileri dünya çapındaki depremleri içeren bir veri setine uygulayacaksınız.

*# modules we'll use*

import pandas as pd

import numpy as np

import seaborn as sns

import datetime

*# read in our data*

landslides = pd.read\_csv("../input/landslide-events/catalog.csv")

*# set seed for reproducibility*

np.random.seed(0)

Şimdi bazı tarihlere bakmaya hazırız!

# Check the data type of our date column[¶](https://www.kaggle.com/code/alexisbcook/parsing-dates#Check-the-data-type-of-our-date-column)

Öncelikle verinin ilk beş satırına bakalım.

landslides.head()

Heyelan veri çerçevesindeki "tarih" sütunuyla çalışacağız. Gerçekten tarih içeriyormuş gibi göründüğünden emin olalım.

*# print the first few rows of the date column*

print(landslides['date'].head())

0 3/2/07

1 3/22/07

2 4/6/07

3 4/14/07

4 4/15/07

Name: date, dtype: object

Evet, bunlar tarih! Ancak bir insan olarak benim bunların tarih olduğunu anlamam, Python'ın da bunların tarih olduğunu bildiği anlamına gelmez. head() çıktısının altında, bu sütunun veri tipinin "object" olduğunu görebilirsiniz.

Pandas, çeşitli veri türlerini saklamak için "object" dtype'ını kullanır, ancak en sık olarak bir sütunda "object" dtype'ını gördüğünüzde, içinde dizeler (string) bulunur.

Buradaki pandas dtype belgelerine bakarsanız, ayrıca belirli bir **datetime64** dtype'ının da olduğunu fark edersiniz. Sütunumuzun dtype'ı datetime64 yerine object olduğu için, Python'ın bu sütunun tarih içerdiğini bilmediğini anlayabiliriz.

İlk birkaç satırı yazdırmadan da sadece bir sütunun dtype'ına bakabiliriz:

*# check the data type of our date column*

landslides['date'].dtype

dtype('O')

Nesnenin veri türüyle harf kodunu eşleştirmek için numpy belgelerini kontrol etmeniz gerekebilir. "O", "nesne" için kullanılan koddur, yani bu iki yöntemin bize aynı bilgiyi verdiğini görebiliriz.

# Convert our date columns to datetime[¶](https://www.kaggle.com/code/alexisbcook/parsing-dates#Convert-our-date-columns-to-datetime)

Tarih sütunumuzun bir tarih olarak tanınmadığını artık bildiğimize göre, onu bir tarih olarak **tanınacak** şekilde dönüştürme zamanı. Bu işlem "tarihleri ayrıştırma" olarak adlandırılır çünkü bir metin alıp onun bileşen kısımlarını belirliyoruz.

Tarihlerimizin hangi biçimde olduğunu, daha fazla bilgiyi bu bağlantıdan bulabileceğiniz **"strftime direktifi"** adlı bir kılavuzla belirleyebiliriz. Temel fikir, tarihin hangi kısımlarının nerede olduğunu ve aralarındaki noktalama işaretlerinin ne olduğunu belirtmeniz gerektiğidir. Bir tarihin **birçok olası bileşeni** vardır, ancak en yaygın olanları gün için **%d**, ay için **%m**, iki basamaklı yıl için **%y** ve dört basamaklı yıl için **%Y**'dir.

Bazı örnekler:

* 1/17/07, "%m/%d/%y" biçimine sahiptir.
* 17-1-2007, "%d-%m-%Y" biçimine sahiptir.

Toprak kayması veri setindeki "date" sütununun başına tekrar baktığımızda, "ay/gün/iki basamaklı yıl" biçiminde olduğunu görebiliriz. Bu nedenle, tarihlerimizi ayrıştırmak için ilk örnektekiyle aynı sözdizimini kullanabiliriz:

*# create a new column, date\_parsed, with the parsed dates*

landslides['date\_parsed'] = pd.to\_datetime(landslides['date'], format="%m/**%d**/%y")

Şimdi yeni sütunun ilk birkaç satırını kontrol ettiğimde, veri tipinin datetime64 olduğunu görebiliyorum. Ayrıca tarihlerimin, varsayılan sıralı datetime nesnelerine (yıl-ay-gün) uyacak şekilde biraz yeniden düzenlendiğini de görebiliyorum.

*# print the first few rows*

landslides['date\_parsed'].head()

0 2007-03-02

1 2007-03-22

2 2007-04-06

3 2007-04-14

4 2007-04-15

Name: date\_parsed, dtype: datetime64[ns]

Artık tarihlerimiz doğru şekilde ayrıştırıldığına göre, onlarla faydalı şekillerde etkileşime geçebiliriz.

Birçok tarih biçimiyle karşılaşırsam ne olur? Burada tarih biçimini biz belirtsek de, bazen tek bir sütunda birden fazla tarih biçimi olduğunda bir hatayla karşılaşabilirsiniz. Böyle bir durumda, pandas'ın doğru tarih biçimini tahmin etmesini sağlayabilirsiniz. Bunu şu şekilde yapabilirsiniz:

landslides['date\_parsed'] = pd.to\_datetime(landslides['Date'], infer\_datetime\_format=True)

**Neden her zaman infer\_datetime\_format = True kullanmıyoruz?** Pandas'ın zaman biçimini tahmin etmesini istememenin iki büyük nedeni var. Birincisi, özellikle biri veri girişi konusunda yaratıcı davrandıysa, pandas her zaman doğru tarih biçimini bulamayabilir. İkincisi ise, tarihlerin tam biçimini belirtmekten çok daha yavaştır.

# Select the day of the month[¶](https://www.kaggle.com/code/alexisbcook/parsing-dates#Select-the-day-of-the-month)

Artık ayrıştırılmış tarihlerden oluşan bir sütuna sahip olduğumuza göre, heyelanın meydana geldiği ayın günü gibi bilgileri çıkarabiliriz.

*# get the day of the month from the date\_parsed column*

day\_of\_month\_landslides = landslides['date\_parsed'].dt.day

day\_of\_month\_landslides.head()

0 2.0

1 22.0

2 6.0

3 14.0

4 15.0

Name: date\_parsed, dtype: float64

Orijinal "date" sütunundan aynı bilgiyi almaya çalışsaydık, şöyle bir hata alırdık: AttributeError: Can only use .dt accessor with datetimelike values. Bunun nedeni, **dt.day**'in "object" dtype'ına sahip bir sütunla nasıl başa çıkacağını bilememesidir. Veri çerçevemizde tarihler olmasına rağmen, onlarla faydalı bir şekilde etkileşime geçebilmemiz için önce onları ayrıştırmamız gerekir.

# Plot the day of the month to check the date parsing[¶](https://www.kaggle.com/code/alexisbcook/parsing-dates#Plot-the-day-of-the-month-to-check-the-date-parsing)

Tarihleri ayrıştırırken en büyük tehlikelerden biri, ay ve günleri karıştırmaktır. to\_datetime() fonksiyonunun çok faydalı hata mesajları olsa da, çıkardığımız ayın günlerinin mantıklı olup olmadığını iki kez kontrol etmek zarar vermez.

Bunu yapmak için, ayın günlerinin bir histogramını çizelim. 1 ile 31 arasında değerler bekliyoruz ve toprak kaymalarının ayın bazı günlerinde diğerlerinden daha yaygın olduğunu varsaymak için bir neden olmadığından, nispeten eşit bir dağılım bekliyoruz. (31'de bir düşüşle, çünkü her ay 31 gün değildir.) Bakalım durum böyle mi:

*# remove na's*

day\_of\_month\_landslides = day\_of\_month\_landslides.dropna()

*# plot the day of the month*

sns.distplot(day\_of\_month\_landslides, kde=False, bins=31)

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

"""

<AxesSubplot:xlabel='date\_parsed'>

Evet, tarihlerimizi doğru şekilde ayrıştırdığımız anlaşılıyor ve bu grafik bana mantıklı geliyor